DATA VISUALISATION USING MATPLOTLIB IN PYTHON FOR DATA ANALYTICS - BY RUPAM GUPTA

Topics Covered :- Intro, syntax, plotting, annotation, subplotting, Sklearn dataset for Histogram, Scatter plots, Seaborn for Heat Maps, plt for Pie Charts

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

•Create publication quality plots.

•Make interactive figures that can zoom, pan, update.

•Customize visual style and layout.

•Export to many file formats.

•Embed in JupyterLab and Graphical User Interfaces.

•Use a rich array of third-party packages built on Matplotlib.

What is Matplotlib?

Matplotlib is a low level graph plotting library in python that serves as a visualization utility.

Matplotlib was created by John D. Hunter.

Matplotlib is open source and we can use it freely.

Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.

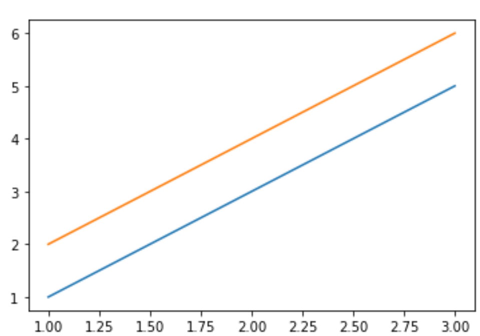
Pyplot

Most of the Matplotlib utilities lies under the pyplot submodule, and are usually imported under the plt alias

In [1]:
```
#import numpy for generating random numbers
import numpy as np

#import matplotlib library
import matplotlib.pyplot as plt

#for grid stylying
from matplotlib import style

#for displaying plot within jupyter notebook
%matplotlib inline
```

In [2]:
```
##direct plotting
x = [1, 2, 3]
y = np.array([[1, 2], [3, 4], [5, 6]])
plt.plot(x, y)
```

Out[2]:
```
[<matplotlib.lines.Line2D at 0x1faf7f77040>,
 <matplotlib.lines.Line2D at 0x1faf7f77070>]
```



In [3]:
```
#generating random numbers(total 10)

#define the dataset, then print the output on the next line.
randomNumber = np.random.rand(10) #np=numpy
```

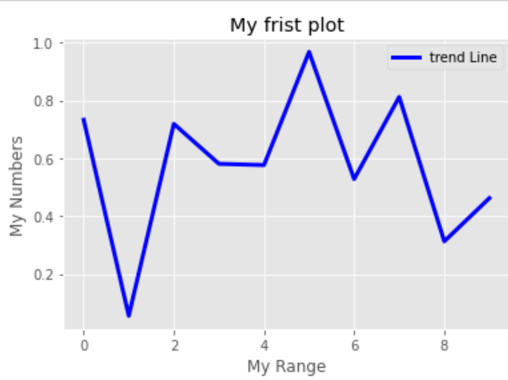
In [4]:
```
print(randomNumber)
```

```
[0.73313153 0.05544482 0.71898401 0.58088385 0.57652829 0.96786767
 0.52813255 0.81220236 0.31288514 0.46216196]
```

NOW WE WILL PLOT THE GRAPH, TO DO SO LETS DEFINE THE FOLLLOWING:-

•STYLE/ PLOT THE RANDOM NUMBERS/ ASSIGN X & Y AXIS LABELS/ TITLE OF PLOT/ LEGEND/ GRAPH NAME/ COMMAND TO SHOW.

In [5]:
```
# STEP=1, there are many styles of plot available in Matplotlib, here we're selecting "ggplot", syntax:-
style.use('ggplot')

#STEP=2, plotting the above variable, setting up color, legend & line width
plt.plot(randomNumber, color= 'b', label = "trend Line", linewidth = 3)

#STEP=3, Naming the X-axis
plt.xlabel('My Range')

#STEP=4, Naming the Y-axis
plt.ylabel('My Numbers')

#STEP=5, title of the plot/graph
plt.title('My frist plot')

#STEP=6, to show the Legend
plt.legend()

#STEP=7, final step to plot the output
plt.show()
```

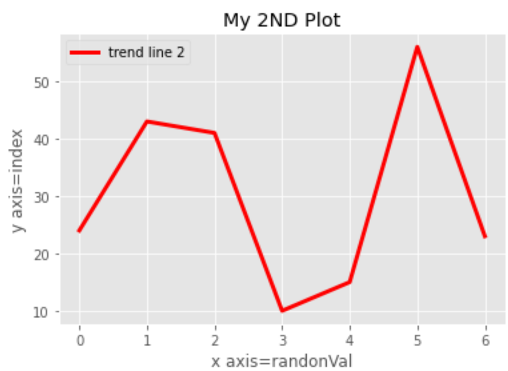


In [6]:
```
#eg=2
my_var=np.array([24,43,41,10,15,56,23])
print(my_var)
```

```
[24 43 41 10 15 56 23]
```

In [7]:
```
style.use('ggplot')
plt.plot(my_var,color="r",label="trend line 2", linewidth=3)
plt.xlabel('x axis=randonVal')
plt.ylabel('y axis=index')
plt.title('My 2ND Plot')
plt.legend()
plt.show
```

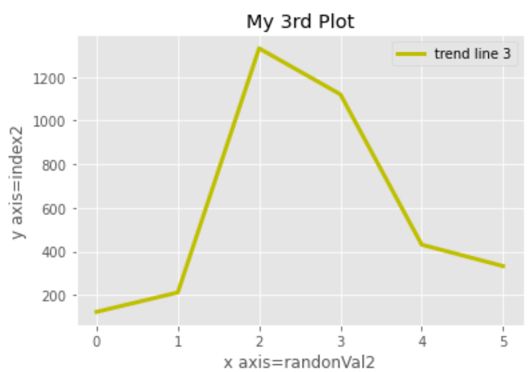
Out[7]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [8]:
```
#eg=3
my_var2=(123,212,1331,1120,431,333) #tuple
print(my_var2)
```

```
(123, 212, 1331, 1120, 431, 333)
```

In [9]:
```
style.use('ggplot')
plt.plot(my_var2,color="y",label="trend line 3", linewidth=3)
plt.xlabel('x axis=randonVal2')
plt.ylabel('y axis=index2')
plt.title('My 3rd Plot')
plt.legend()
plt.show()
```



Let's see how to plot multiple graphs using Matplotlib in python

In [10]:
```
#website traffic data.
#no of users/visitors on the website.
web_monday = [123,645,950,290,1630,1450,1034,295,465,205,80]

web_tuesday = [95,680,889,1145,1670,1323,1119,1265,510,310,110]

web_wednesday = [105,630,700,1006,1520,1124,1239,1380,580,610,230]

#time distribution->hourly.
time_hrs = [7,8,9,10,11,12,13,14,15,16,17]

print(web_monday)
print(web_tuesday)
print(web_wednesday)
print(time_hrs)
```

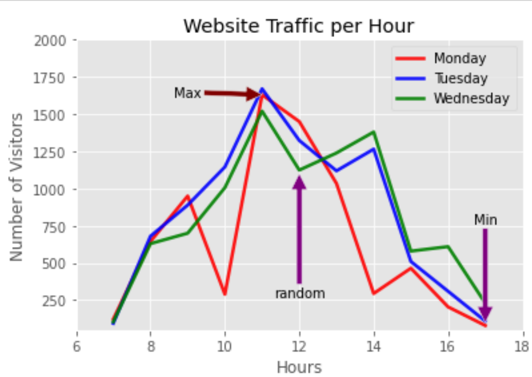
```
[123, 645, 950, 290, 1630, 1450, 1034, 295, 465, 205, 80]
[95, 680, 889, 1145, 1670, 1323, 1119, 1265, 510, 310, 110]
[105, 630, 700, 1006, 1520, 1124, 1239, 1380, 580, 610, 230]
[7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]
```

In [11]:

```
#selecting the plot style. Here we're using ggplot
style.use("ggplot")

#plot the website traffic data by taking X-axis as time_hrs and Y-axis as web_customers.
#alpha is an attribute which controls the transparency(light/dark) of the line, Range(0-1).
#more the alpha value,more transparent the line is.

#plotting for Monday traffic.
plt.plot(time_hrs,web_monday,color = 'r',label = 'Monday',linewidth = 2.5,alpha = 0.9)

#plotting for Tuesday traffic.
plt.plot(time_hrs,web_tuesday,color = 'b',label = 'Tuesday',linewidth = 2.5,alpha = 0.9)

#plotting for Wednesday traffic.
plt.plot(time_hrs,web_wednesday,color = 'g',label = 'Wednesday',linewidth = 2.5,alpha = 0.9)

#set range for both x & y axis (x,x,y,y)
plt.axis([6,18,50,2000])

#Naming x-axis as Hours
plt.xlabel('Hours')

#Naming y-axis as Traffic(no. of visitors)
plt.ylabel('Number of Visitors')

#Title of the plot
plt.title('Website Traffic per Hour')

#enable legend
plt.legend()

#show the plot
plt.show()
```

Lets assign MAX, MIN & Random annotation on the above graph

In [12]:
```python
#selecting the plot style. Here we're using ggplot
style.use("ggplot")

#plot the website traffic data by taking X-axis as time_hrs and Y-axis as web_customers.
#alpha is an attribute which controls the transparency(light/dark) of the line, Range(0-1).
#more the alpha value,more transparent the line is.

#plotting for Monday traffic.
plt.plot(time_hrs,web_monday,color = 'r',label = 'Monday',linewidth = 2.5,alpha = 0.9)

#plotting for Tuesday traffic.
plt.plot(time_hrs,web_tuesday,color = 'b',label = 'Tuesday',linewidth = 2.5,alpha = 0.9)

#plotting for Wednesday traffic.
plt.plot(time_hrs,web_wednesday,color = 'g',label = 'Wednesday',linewidth = 2.5,alpha = 0.9)

#set range for both x & y axis (x,x,y,y)
plt.axis([6,18,50,2000])

#Naming x-axis as Hours
plt.xlabel('Hours')

#Naming y-axis as Traffic(no. of visitors)
plt.ylabel('Number of Visitors')

#Title of the plot
plt.title('Website Traffic per Hour')

#.annotate
#Max-> Annotation Text
#ha & va-> horizontal and vertical alignment respectively
#xytext-> text position
#xy -> indicates the arrow position
#arrowprops -> indicates the properties of the arrow

#show me maximum
plt.annotate('Max',ha = 'center',va = 'bottom',xytext = (9,1600),xy = (11,1630),arrowprops = {'facecolor':'maroon'})

#show me minimum
plt.annotate('Min',ha = 'center',va = 'bottom',xytext = (17,750),xy = (17,100),arrowprops = {'facecolor':'purple'})

#trying to show some random dip.
plt.annotate('random',ha = 'center',va = 'bottom',xytext = (12,250),xy = (12,1100),arrowprops = {'facecolor':'purple'})

#enable legend
plt.legend()

#show the plot
plt.show()
```

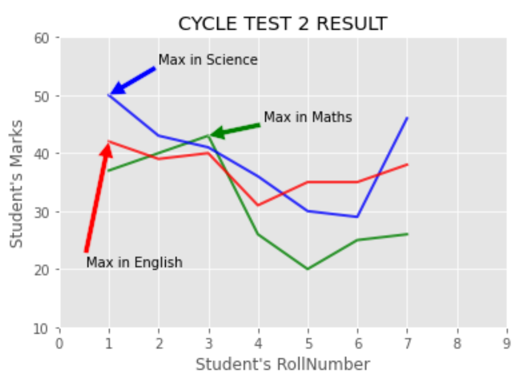


Lets try 1 more exp to understand the it better. take 4 random lists or tuples and perform the above steps.

In [13]:
```python
marksMaths=[37,40,43,26,20,25,26]
marksScience=[50,43,41,36,30,29,46]
marksEnglish=[42,39,40,31,35,35,38]

rollnumber=[1,2,3,4,5,6,7]

print(marksMaths)
print(marksScience)
print(marksEnglish)
print(rollnumber)
```

```
[37, 40, 43, 26, 20, 25, 26]
[50, 43, 41, 36, 30, 29, 46]
[42, 39, 40, 31, 35, 35, 38]
[1, 2, 3, 4, 5, 6, 7]
```

In [71]:
```python
style.use("ggplot")
plt.plot(rollnumber, marksMaths, color="green", label="Maths", linewidth=2, alpha=0.8)
plt.plot(rollnumber, marksScience, color="blue", label="Science", linewidth=2, alpha=0.8)
plt.plot(rollnumber, marksEnglish, color="red", label="English", linewidth=2, alpha=0.8)
plt.axis([0,9,10,60])
plt.xlabel("Student's RollNumber")
plt.ylabel("Student's Marks")
plt.title("CYCLE TEST 2 RESULT")

plt.annotate("Max in Science", ha="center", va="bottom", xytext= (3,55), xy=(1,50),arrowprops = {'facecolor':'blue'} )

plt.annotate("Max in English", ha="center", va="bottom", xytext= (1.5,20), xy=(1,42),arrowprops = {'facecolor':'red'} )

plt.annotate("Max in Maths", ha="center", va="bottom", xytext= (5,45), xy=(3,43),arrowprops = {'facecolor':'green'})
plt.show()
```



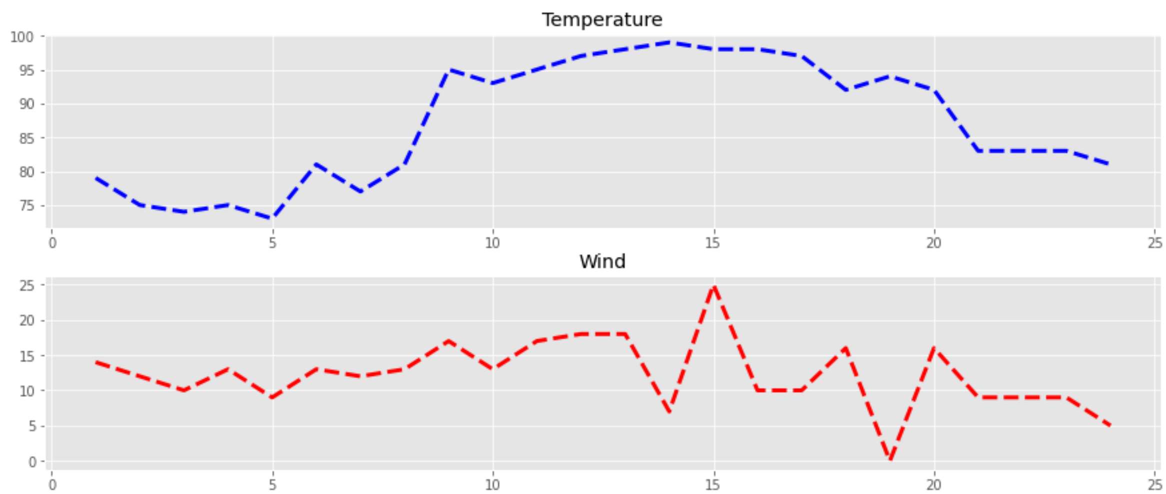
Matplotlib Subplotting

define temp,wind,humidity,precipitation data and Time hrs data

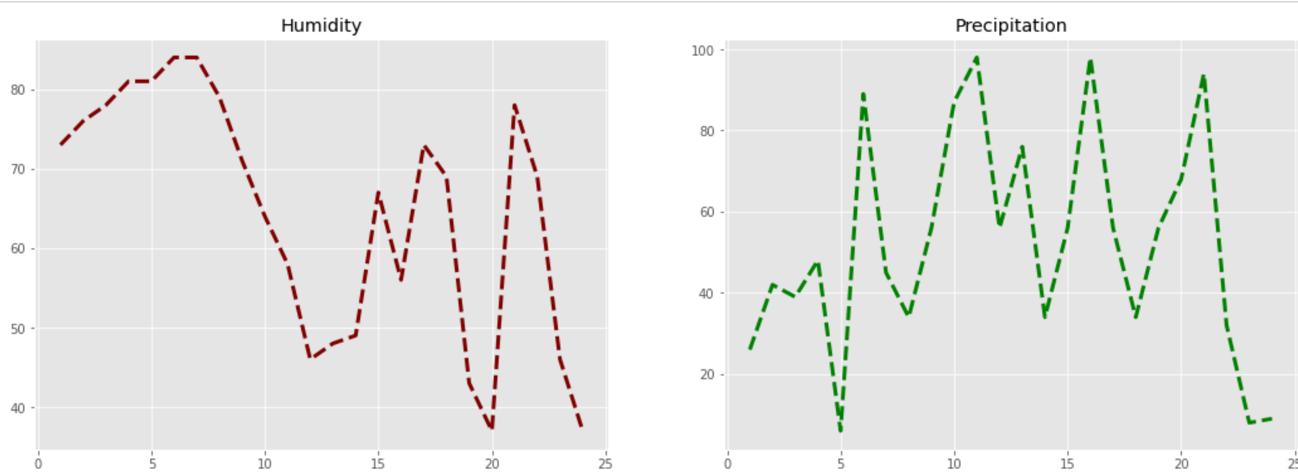
In [28]:
```python
temp_data = [79,75,74,75,73,81,77,81,95,93,95,97,98,99,98,98,97,92,94,92,83,83,83,81]
wind_data = [14,12,10,13,9,13,12,13,17,13,17,18,18,7,25,10,10,16,0,16,9,9,9,5]
humidity_data = [73,76,78,81,81,84,84,79,71,64,58,46,48,49,67,56,73,69,43,37,78,69,46,37]
precip_data = [26,42,39,48,6,89,45,34,56,87,98,56,76,34,56,98,56,34,56,68,94,32,8,9]

time_hrs = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24]
```

```
**draw subplots for (1,2,1) and (1,2,2)**

Here (1,2,1) denotes (rows,columns & index)
```

In [76]:
```
#defining the plot size
plt.figure(figsize=(15,6))##length and breadth of figure

#adjust the horizontal space between two graphs.
plt.subplots_adjust(hspace = 0.25)

#defining the plot location(rows,columns & index)
plt.subplot(2,1,1)
plt.title('Temperature')
plt.plot(time_hrs,temp_data,color = 'b',linestyle = '--',linewidth = 3) #here in linestyle '--' dashed line is used

#defining the plot location(rows,columns & index)
plt.subplot(2,1,2)
plt.title('Wind')
plt.plot(time_hrs,wind_data,color = 'r',linestyle = '--',linewidth = 3) #here in linestyle '--' dashed line is used

plt.show()
```

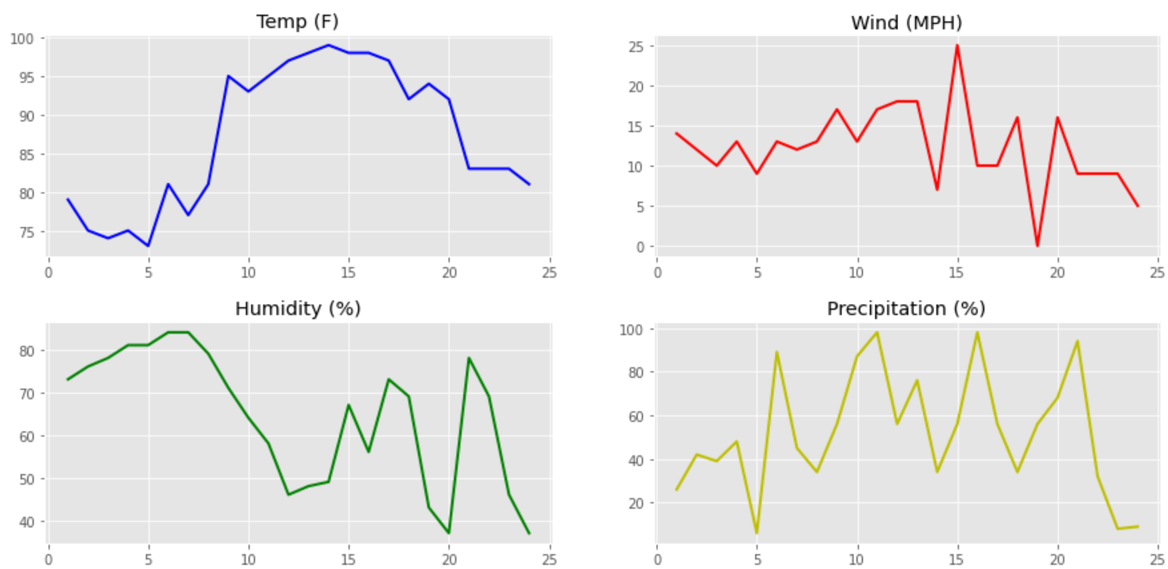


In [85]:
```
# draw subplots for (1,2,1) and (1,2,2)
plt.figure(figsize=(18,6))
plt.subplots_adjust(hspace = 0.25)
plt.subplot(1,2,1)
plt.title('Humidity')
plt.plot(time_hrs,humidity_data,color = 'maroon',linestyle = '--',linewidth = 3)

plt.subplot(1,2,2)
plt.title('Precipitation')
plt.plot(time_hrs,precip_data,color = 'green',linestyle = '--',linewidth = 3)

plt.show()
```



Compilling 4 plots together

In [90]:
```
# draw subplots for (2,2,1),(2,2,2),(2,2,3),(2,2,4)
plt.figure(figsize=(15,7))

#space between 2 graphs
plt.subplots_adjust(hspace = 0.3)

plt.subplot(2,2,1)
plt.title('Temp (F)')
plt.plot(time_hrs,temp_data,color = 'b',linestyle = '-',linewidth = 2)

plt.subplot(2,2,2)
plt.title('Wind (MPH)')
plt.plot(time_hrs,wind_data,color = 'r',linestyle = '-',linewidth = 2)

plt.subplot(2,2,3)
plt.title('Humidity (%)')
plt.plot(time_hrs,humidity_data,color = 'g',linestyle = '-',linewidth = 2)

plt.subplot(2,2,4)
plt.title('Precipitation (%)')
plt.plot(time_hrs,precip_data,color = 'y',linestyle = '-',linewidth = 2)

plt.show()
```



Histogram, Scatter plots, Heat Maps, Pie Charts

Import the boston dataset from sklearn library

In [91]:
```
from sklearn.datasets import load_boston

import matplotlib.pyplot as plt
from matplotlib import style
%matplotlib inline
```

In [92]:
```
import sklearn.datasets as skd
```

In [93]:
```
# Load boston dataset
boston_data = load_boston()
```

In [94]:
```python
#view boston dataset
print(boston_data.DESCR)
```

```
.. _boston_dataset:

Boston house prices dataset
---------------------------

**Data Set Characteristics:**

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is usually the target.

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of black people by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's

    :Missing Attribute Values: None

    :Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.
https://archive.ics.uci.edu/ml/machine-learning-databases/housing/ (https://archive.ics.uci.edu/ml/machine-learning-databases/h
ousing/)


This dataset was taken from the StatLib library which is maintained at Carnegie Mellon University.

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic
prices and the demand for clean air', J. Environ. Economics & Management,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnostics
...', Wiley, 1980.   N.B. Various transformations are used in the table on
pages 244-261 of the latter.

The Boston house-price data has been used in many machine learning papers that address regression
problems.

.. topic:: References

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential Data and Sources of Collinearity', Wiley, 1980. 24
4-261.
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In Proceedings on the Tenth International Conference
of Machine Learning, 236-243, University of Massachusetts, Amherst. Morgan Kaufmann.
```

In [95]:
```python
# define x-axis for the data
x_axis = boston_data.data

# define y-axis for the data
y_axis = boston_data.target
```

In [96]:
```python
x_axis
```

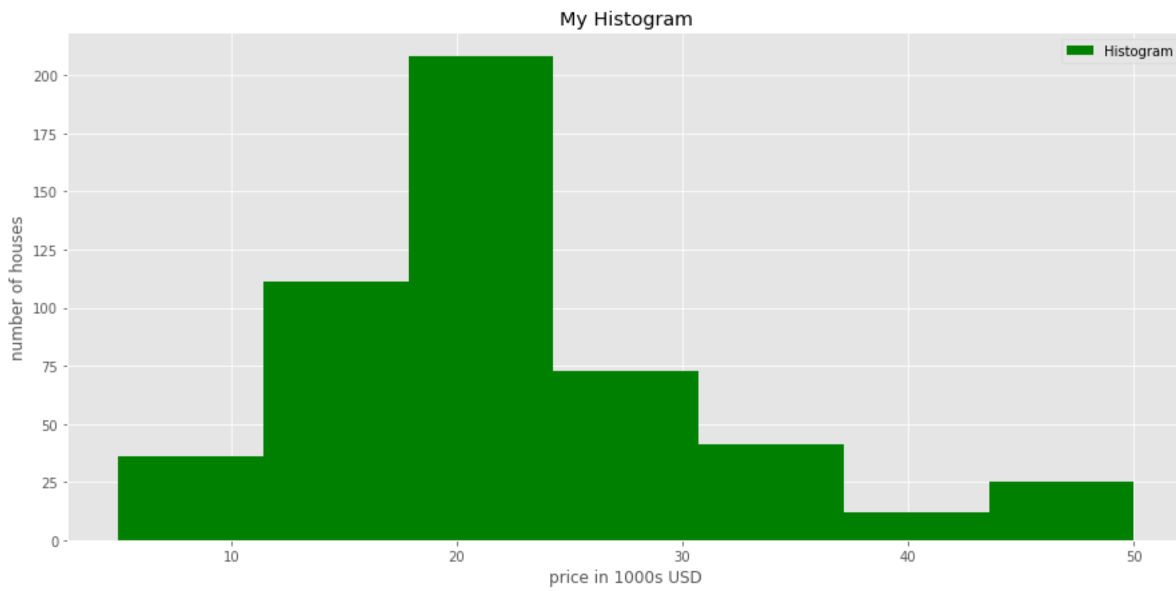
Out[96]:
```
array([[6.3200e-03, 1.8000e+01, 2.3100e+00, ..., 1.5300e+01, 3.9690e+02,
        4.9800e+00],
       [2.7310e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9690e+02,
        9.1400e+00],
       [2.7290e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9283e+02,
        4.0300e+00],
       ...,
       [6.0760e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
        5.6400e+00],
       [1.0959e-01, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9345e+02,
        6.4800e+00],
       [4.7410e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
        7.8800e+00]])
```

In [97]:
```
y_axis
```

Out[97]:
```
array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15. ,
       18.9, 21.7, 20.4, 18.2, 19.9, 23.1, 17.5, 20.2, 18.2, 13.6, 19.6,
       15.2, 14.5, 15.6, 13.9, 16.6, 14.8, 18.4, 21. , 12.7, 14.5, 13.2,
       13.1, 13.5, 18.9, 20. , 21. , 24.7, 30.8, 34.9, 26.6, 25.3, 24.7,
       21.2, 19.3, 20. , 16.6, 14.4, 19.4, 19.7, 20.5, 25. , 23.4, 18.9,
       35.4, 24.7, 31.6, 23.3, 19.6, 18.7, 16. , 22.2, 25. , 33. , 23.5,
       19.4, 22. , 17.4, 20.9, 24.2, 21.7, 22.8, 23.4, 24.1, 21.4, 20. ,
       20.8, 21.2, 20.3, 28. , 23.9, 24.8, 22.9, 23.9, 26.6, 22.5, 22.2,
       23.6, 28.7, 22.6, 22. , 22.9, 25. , 20.6, 28.4, 21.4, 38.7, 43.8,
       33.2, 27.5, 26.5, 18.6, 19.3, 20.1, 19.5, 19.5, 20.4, 19.8, 19.4,
       21.7, 22.8, 18.8, 18.7, 18.5, 18.3, 21.2, 19.2, 20.4, 19.3, 22. ,
       20.3, 20.5, 17.3, 18.8, 21.4, 15.7, 16.2, 18. , 14.3, 19.2, 19.6,
       23. , 18.4, 15.6, 18.1, 17.4, 17.1, 13.3, 17.8, 14. , 14.4, 13.4,
       15.6, 11.8, 13.8, 15.6, 14.6, 17.8, 15.4, 21.5, 19.6, 15.3, 19.4,
       17. , 15.6, 13.1, 41.3, 24.3, 23.3, 27. , 50. , 50. , 50. , 22.7,
       25. , 50. , 23.8, 23.8, 22.3, 17.4, 19.1, 23.1, 23.6, 22.6, 29.4,
       23.2, 24.6, 29.9, 37.2, 39.8, 36.2, 37.9, 32.5, 26.4, 29.6, 50. ,
       32. , 29.8, 34.9, 37. , 30.5, 36.4, 31.1, 29.1, 50. , 33.3, 30.3,
       34.6, 34.9, 32.9, 24.1, 42.3, 48.5, 50. , 22.6, 24.4, 22.5, 24.4,
       20. , 21.7, 19.3, 22.4, 28.1, 23.7, 25. , 23.3, 28.7, 21.5, 23. ,
       26.7, 21.7, 27.5, 30.1, 44.8, 50. , 37.6, 31.6, 46.7, 31.5, 24.3,
       31.7, 41.7, 48.3, 29. , 24. , 25.1, 31.5, 23.7, 23.3, 22. , 20.1,
       22.2, 23.7, 17.6, 18.5, 24.3, 20.5, 24.5, 26.2, 24.4, 24.8, 29.6,
       42.8, 21.9, 20.9, 44. , 50. , 36. , 30.1, 33.8, 43.1, 48.8, 31. ,
       36.5, 22.8, 30.7, 50. , 43.5, 20.7, 21.1, 25.2, 24.4, 35.2, 32.4,
       32. , 33.2, 33.1, 29.1, 35.1, 45.4, 35.4, 46. , 50. , 32.2, 22. ,
       20.1, 23.2, 22.3, 24.8, 28.5, 37.3, 27.9, 23.9, 21.7, 28.6, 27.1,
       20.3, 22.5, 29. , 24.8, 22. , 26.4, 33.1, 36.1, 28.4, 33.4, 28.2,
       22.8, 20.3, 16.1, 22.1, 19.4, 21.6, 23.8, 16.2, 17.8, 19.8, 23.1,
       21. , 23.8, 23.1, 20.4, 18.5, 25. , 24.6, 23. , 22.2, 19.3, 22.6,
       19.8, 17.1, 19.4, 22.2, 20.7, 21.1, 19.5, 18.5, 20.6, 19. , 18.7,
       32.7, 16.5, 23.9, 31.2, 17.5, 17.2, 23.1, 24.5, 26.6, 22.9, 24.1,
       18.6, 30.1, 18.2, 20.6, 17.8, 21.7, 22.7, 22.6, 25. , 19.9, 20.8,
       16.8, 21.9, 27.5, 21.9, 23.1, 50. , 50. , 50. , 50. , 50. , 13.8,
       13.8, 15. , 13.9, 13.3, 13.1, 10.2, 10.4, 10.9, 11.3, 12.3,  8.8,
        7.2, 10.5,  7.4, 10.2, 11.5, 15.1, 23.2,  9.7, 13.8, 12.7, 13.1,
       12.5,  8.5,  5. ,  6.3,  5.6,  7.2, 12.1,  8.3,  8.5,  5. , 11.9,
       27.9, 17.2, 27.5, 15. , 17.2, 17.9, 16.3,  7. ,  7.2,  7.5, 10.4,
        8.8,  8.4, 16.7, 14.2, 20.8, 13.4, 11.7,  8.3, 10.2, 10.9, 11. ,
        9.5, 14.5, 14.1, 16.1, 14.3, 11.7, 13.4,  9.6,  8.7,  8.4, 12.8,
       10.5, 17.1, 18.4, 15.4, 10.8, 11.8, 14.9, 12.6, 14.1, 13. , 13.4,
       15.2, 16.1, 17.8, 14.9, 14.1, 12.7, 13.5, 14.9, 20. , 16.4, 17.7,
       19.5, 20.2, 21.4, 19.9, 19. , 19.1, 19.1, 20.1, 19.9, 19.6, 23.2,
       29.8, 13.8, 13.3, 16.7, 12. , 14.6, 21.4, 23. , 23.7, 25. , 21.8,
       20.6, 21.2, 19.1, 20.6, 15.2,  7. ,  8.1, 13.6, 20.1, 21.8, 24.5,
       23.1, 19.7, 18.3, 21.2, 17.5, 16.8, 22.4, 20.6, 23.9, 22. , 11.9])
```

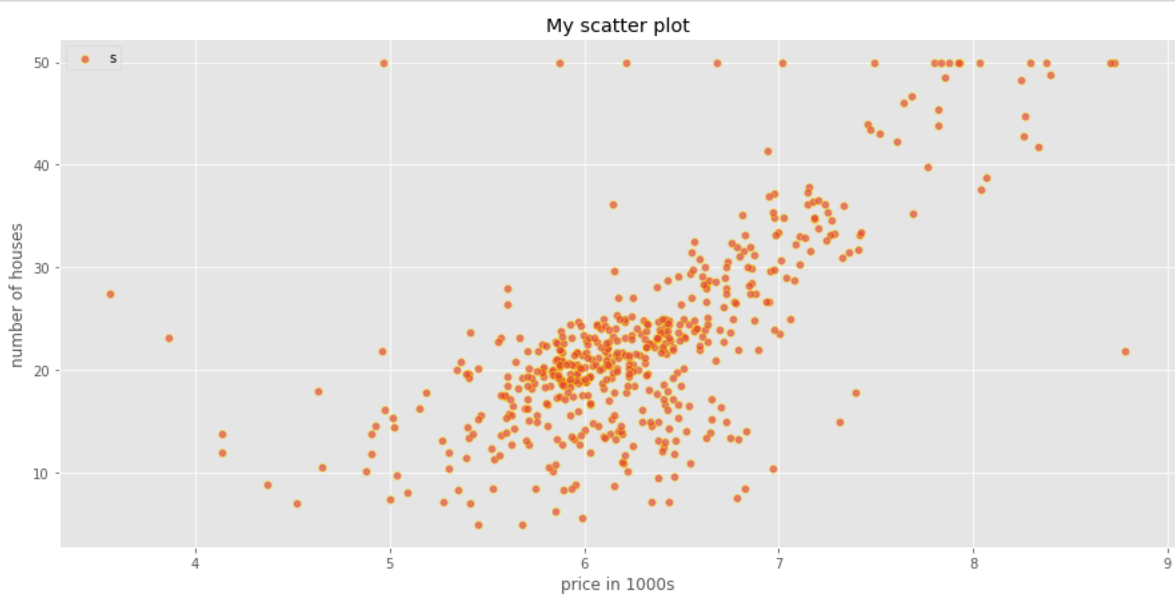
PLOTTING HISTOGRAM

In [105]:
```
style.use('ggplot')

plt.figure(figsize = (15,7))

##'bar', 'barstacked', 'step', 'stepfilled'
plt.hist(y_axis,bins = 7,histtype='barstacked',align='mid',orientation='vertical',color='green',label='Histogram',stacked=True)

plt.xlabel('price in 1000s USD')
plt.ylabel('number of houses')

plt.title("My Histogram")
plt.legend()

plt.show()
```



Plotting Scatter Plot

In [123]:
```
style.use('ggplot')

plt.figure(figsize = (15,7))

plt.scatter(boston_data.data[:,5],boston_data.target,alpha=0.7,linewidths=0.5,edgecolors='Yellow',plotnonfinite= "true")

plt.xlabel('price in 1000s')
plt.ylabel('number of houses')
plt.legend("size")
plt.title("My scatter plot")
plt.show()
```



Plotting Heat Map

In [124]:
```python
# import matplot library
import matplotlib.pyplot as plt

# import seaborn library
import seaborn as sns
# to show plot on notebook
%matplotlib inline
```

Load flight data from the sns dataset

In [125]:
```python
flight_data = sns.load_dataset('flights')
```

In [126]:
```python
#view top 5 records
flight_data.head()
```

Out[126]:

	year	month	passengers
0	1949	Jan	112
1	1949	Feb	118
2	1949	Mar	132
3	1949	Apr	129
4	1949	May	121

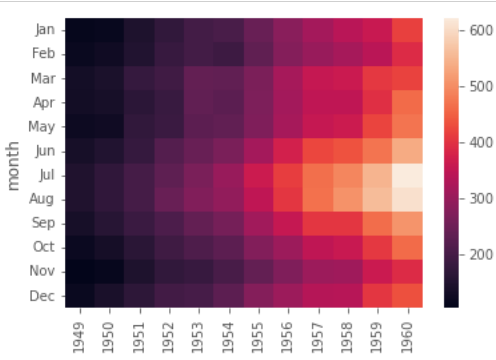
In [127]:
```python
#use pivot method to arrange the datasets
flight_data = flight_data.pivot('month','year','passengers')
```

In [128]:
```python
#view the datasets
flight_data
```

Out[128]:

year	1949	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	1960
month												
Jan	112	115	145	171	196	204	242	284	315	340	360	417
Feb	118	126	150	180	196	188	233	277	301	318	342	391
Mar	132	141	178	193	236	235	267	317	356	362	406	419
Apr	129	135	163	181	235	227	269	313	348	348	396	461
May	121	125	172	183	229	234	270	318	355	363	420	472
Jun	135	149	178	218	243	264	315	374	422	435	472	535
Jul	148	170	199	230	264	302	364	413	465	491	548	622
Aug	148	170	199	242	272	293	347	405	467	505	559	606
Sep	136	158	184	209	237	259	312	355	404	404	463	508
Oct	119	133	162	191	211	229	274	306	347	359	407	461
Nov	104	114	146	172	180	203	237	271	305	310	362	390
Dec	118	140	166	194	201	229	278	306	336	337	405	432

In [134]:
```python
# use heatmap method o generate the heatmap of the flights data
sns.heatmap(flight_data)
sns.xlabel='years'
sns.ylabel='months'
```



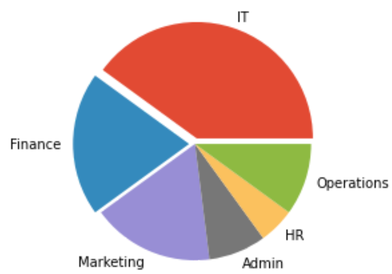
Plotting Pie Charts

In [136]:

```
# import matplot library
import matplotlib.pyplot as plt
# to show plot on notebook
%matplotlib inline
```

In [138]:

```

# job data in percentile
job_data = ['40','20','17','8','5','10']

# define label as different departments
depart = ['IT','Finance','Marketing','Admin','HR','Operations']

# explode the first slice which is IT
explode = (0.05,0.05,0,0,0,0)

# draw the piechart and set the parameters
plt.pie(job_data,labels=depart,explode=explode)

# show the plot
plt.show()
```



In [ ]:

```
```